Alejandro Ossio Díaz, A01209122.

Gerardo Daniel Naranjo Gallegos, A01209499.

Roberto Saavedra Figueroa, A01209689.

Profesor: Agustín Domínguez Oviedo.

Curso de microcontroladores.

ITESM, campus Querétaro.

28 de marzo de 2016.

**Práctica 2: Frecuencímetro digital.**

# Descripción de la práctica.

Esta práctica pretende programar un microcontrolador ATMega328P y crear un circuito, en donde se implemente un frecuencímetro digital. Primero, probando el código de manera virtual en Proteus, para poder generar un segundo exacto; después, probándolo físicamente en el circuito, haciendo uso del oscilador interno del microcontrolador (8 MHz); posteriormente, probar el mismo código utilizando un cristal de cuarzo (8 MHz); finalmente, calibrar el código para obtener el dato exacto con el cristal de cuarzo, aún en frecuencias altas (desde 0 Hz hasta 2 MHz).

Los componentes utilizados en el circuito son: un microcontrolador ATMega328P, un cristal de cuarzo (8 MHz), dos capacitores de 22 pF, un potenciómetro y un LCD. También se utilizan *displays* de siete segmentos en Proteus, pero no en el circuito físico; son utilizados para ver la frecuencia dada.

El microcontrolador es programado por medio de la computadora, con la ayuda del software Atmel Studio y con el código, que se puede encontrar posteriormente en el documento. Este chip, al ser un microcontrolador, se encargará de llevar a cabo todo el procesamiento necesario para cumplir con el código; el objetivo es crear el frecuencímetro. Se utiliza un cristal de cuarzo de 8 MHz por tener una mayor precisión que el oscilador interno del microcontrolador.

El potenciómetro funciona regulando el brillo del LCD. El potenciómetro rotatorio funciona como una resistencia variable o, mejor dicho, que puede ser regulada por el usuario, a través de una perilla. El potenciómetro manda un valor entre 0 y 1023 (en función del movimiento de la perilla), dato que será interpretado por el microprocesador y que determina el nivel de brillo del LCD.

# Objetivo.

Nuestro objetivo es lograr tener un retardo de un segundo exacto con la simulación en Proteus. Una vez conseguido, probamos en el circuito físico para registrar el porcentaje de error obtenido al hacer uso del oscilador interno del microcontrolador y, después, probarlo con un cristal de cuarzo; ambos cuentan con una frecuencia de operación de 8 MHz, por lo tanto, al tener el porcentaje de error de ambos, nos podremos dar cuenta que el oscilador interno tiene menor precisión.

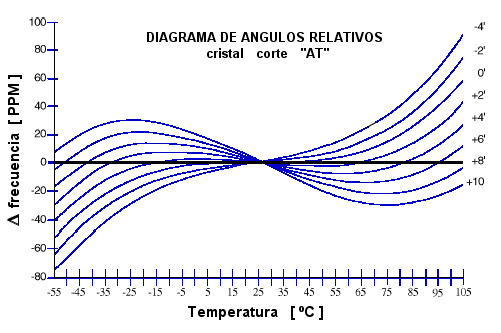
# Marco teórico.

**Investigar cuál es la frecuencia máxima que puede tener una señal externa para que un *timer* del AVR contando eventos sea capaz de contar todos y cada uno de sus flancos.**

La frecuencia máxima que puede tener una señal externa es la mitad de la frecuencia de la muestra (usando el teorema de Nyquist). No obstante, debido a la variación de la frecuencia de reloj del sistema y el ciclo de trabajo causado por fuentes osciladoras (cristal y capacitores), se recomienda que la frecuencia máxima de una fuente de reloj externa sea menor que fclk\_I/O/2.5.

**Investigar el parámetro PPM de los cristales de cuarzo.**

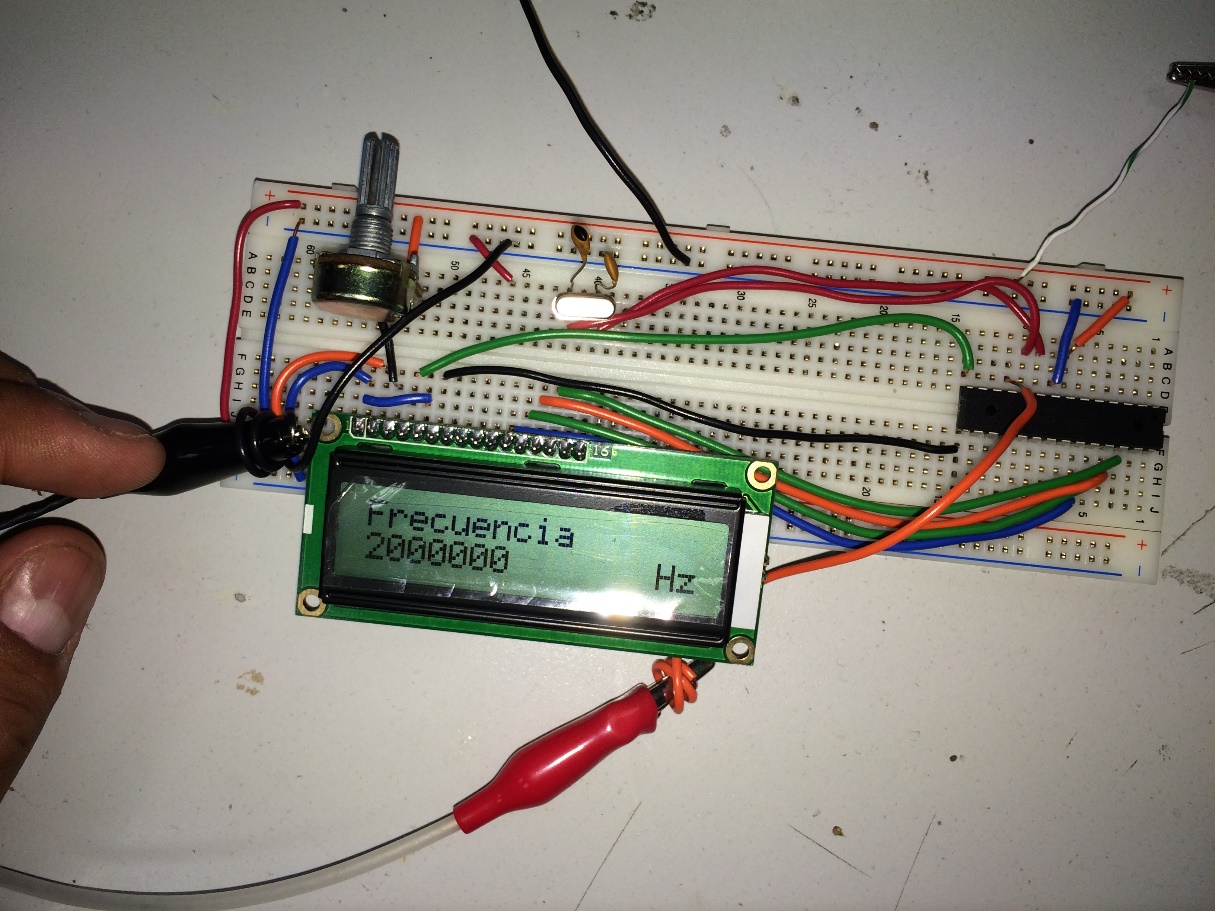
Variación en partes por millón (PPM) con respecto a la temperatura del cristal.

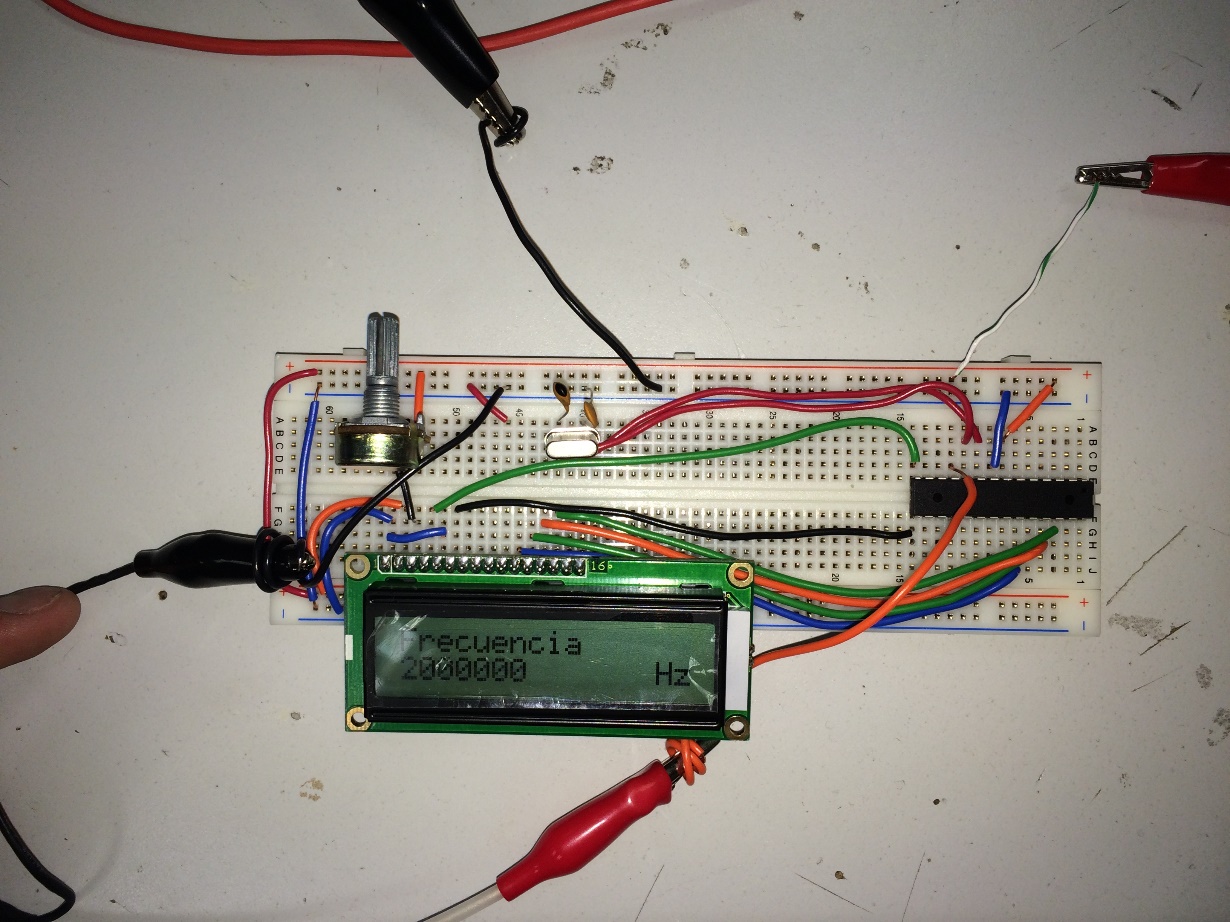


# Diagrama eléctrico completo de la práctica.



# Fotografía del circuito funcionando.





# Código fuente documentado.

//bibliotecas:

#include <io.h>

#include <stdio.h>

#include <delay.h>

#include "display.h"

//declaración de variables:

unsigned long int frec;

unsigned char THH1;

int i;

char res[8];

//inicio de main:

void main() {

//Configuración de puertos:

DDRC=0x0F; //Define al puerto C como entrada en los primeros cuatro pines y como salida en los cuatro pines menos significativos

DDRB=0xFF; //Define a todos los pines del puerto B como salida

//Inicializar LCD:

ConfiguraLCD();

//Inicio ciclo while:

while(1){

//Asignar Timers a 0 y asignar el Timer a 1 a un reloj externo (activo en bajo):

THH1=0;

TCNT1H=0;

TCNT1L=0;

TCCR1B=0x06;

for(i=0; i<50; i++){

//retrasos:

delay\_ms(19); //retrasa 19 milisegundos

delay\_us(976); //retrasa 976 microsegundos

//Inicio ciclo if:

if(TIFR1.TOV1==1){

THH1++; //Incrementa en +1 las cuentas

TIFR1.TOV1=1; //Reiniciar a cero la bandera de overflow

} //Fin ciclo if

//Inicio ciclo else:

else{

//Retrasos

#asm("NOP"); //Retrasa 125 microsegundos

#asm("NOP");

#asm("NOP");

#asm("NOP");

#asm("NOP");

#asm("NOP");

#asm("NOP");

} //Fin ciclo else

} //Fin ciclo for

//Retrasos

#asm("NOP"); //Retrasa 125 microsegundos

#asm("NOP");

#asm("NOP");

TIFR1.TOV1=1;

TCCR1B=0;

//Conversión de cuentas a la frecuencia

frec=TCNT1L;

frec=frec+((long)TCNT1H\*256);

frec=frec+((long)THH1\*65536);

MoverCursor(0,0); //Posición del cursos en LCD (nivel superior)

StringLCD("Frecuencia"); //Imprime “Frecuencia” en el LCD

MoverCursor(0,1); //Posición del cursor en LCD (nivel inferior)

sprintf(res,"%07ld",frec); //Imprime

StringLCDVar(res); //Imprime en LCD el valor de la variable “res”

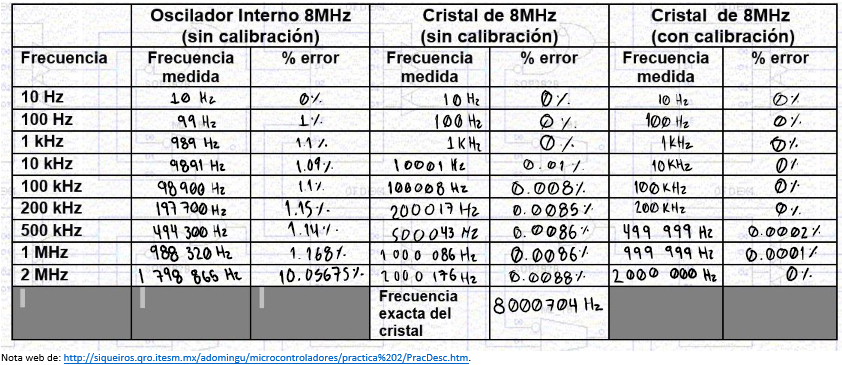
MoverCursor(14,1); //Posición del cursor en LCD (nivel inferior)

StringLCD("Hz"); //Imprime en LCD “Hz”

} //Fin ciclo while

} //Fin de Main

# Tabla de mediciones.



# Conclusiones.

**Gerardo Naranjo:**

Esta práctica resulta tardada por el hecho de ser a prueba y error, con la intención de obtener los datos exactos. Tuvimos dificultades para la conexión entre la laptop y el microcontrolador, por lo que tardamos más tiempo. Fuera de los inconvenientes, logramos observar cómo un cristal de cuarzo es más preciso que el oscilador interno del microcontrolador (que bien puede ser usado, pero atenerse a resultados variables).

**Roberto Figueroa:**

Con la práctica aprendimos a generar retardos más precisos utilizando el cristal de cuarzo líquido, esto nos permitió poder crear un frecuencímetro casi sin errores. Lo más difícil que encontramos a lo largo de la práctica, fue la conexión entre el micro y la computadora, ya que siempre nos ha dado un poco de problemas para que lo lea, una vez pudiéndolo programar, no tuvimos ningún otro problema.

**Alejandro Ossio:**

Con esta práctica se pudo observar que es posible generar retardos mucho más precisos utilizando el modo CTC, lo cual permitió crear un frecuencímetro exacto usando tanto el oscilador interno del microcontrolador como uno externo.

# Referencias

ATMEL. (s.f.). *Datasheet ATMega328P .* Obtenido de 8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash.